

Procedural Analysis of Choice Rules with Applications to Bounded Rationality*

Yuval Salant

Northwestern University

y-salant@northwestern.edu

December 4, 2008

Abstract

I study how limited abilities to process information affect choice behavior. I model the decision making process by an automaton, and measure the complexity of a specific choice rule by the minimal number of categories an automaton implementing the rule uses to process information. I establish that any choice rule that is less complicated than utility maximization displays framing effects. I then prove that the unique choice rule that results from an optimal tradeoff between maximizing utility and minimizing complexity is a history-dependent satisficing procedure that displays primacy and recency effects, default tendency and choice overload.

*I am indebted to Bob Wilson for his devoted guidance, constant support, and most valuable suggestions. I am grateful to Ariel Rubinstein and Jeremy Bulow for the encouragement, productive discussions, and important comments. I thank Gil Kalai, Ron Siegel, and Andy Skrzypacz for most insightful feedback in various stages of this project. I also thank Matt Jackson, Jon Levin, Michael Ostrovsky, Roy Radner, Ilya Segal, and seminar participants at UC Berkeley, Brown, Caltech, Harvard, Hebrew University, Iowa, LSE, MIT, Northwestern, NYU, Stanford, Tel-Aviv, UCL, Washington University in St. Louis, and Yale for helpful comments. This research is supported in part by the Leonard W. and Shirley R. Ely Fellowship of the Stanford Institute for Economic Policy Research.

1 Introduction

Economists have long recognized the need to explore cognitive and procedural aspects of decision making such as limited abilities to process information. Herbert Simon [25] argued that procedural considerations may push decision makers toward various forms of “boundedly” rational behavior, and suggested satisficing as an alternative to utility maximization. Daniel Kahneman and Amos Tversky [13, 26] identified important behavioral biases and heuristics in decision making, and developed decision models that accommodate them. The Bounded Rationality and Behavioral Economics literature has pursued the ideas of Simon and Kahneman and Tversky from different perspectives.¹ A common claim in this literature is that procedural and cognitive considerations lead decision makers to behave in ways that are inconsistent with utility maximization.

This paper studies how limited abilities to process information affect choice behavior. In the model of individual choice I investigate, the decision maker chooses from lists, i.e., sequences of alternatives. For example, in the online marketplace, consumers choose among products listed in order, and in the labor market, recruiters interview candidates sequentially.

To make choices, the decision maker processes the elements of the list sequentially, in the order in which they appear. He does so with the aid of categories that effectively summarize past information relevant for future information processing. For example, a satisficer, who chooses from every list the first alternative that exceeds some aspiration threshold, may categorize the part of the list seen so far according to whether it contains a satisfactory element. He can then process the next alternative in the list based solely on this classification. A rational decision maker, who maximizes a utility function u , may categorize lists according to the identity of the u -best element in each of them, ignoring any other information about the content of the list. These decision makers use categories to effectively classify all the beginnings of lists that are identical for information processing purposes.

Categorizing information is considered by psychologists to be an essential cognitive mechanism that provides “maximum information with the least cognitive effort” (Rosch [19].) Mervis and Rosch [17] define that a “category exists whenever two or more distinguishable objects or events are treated equivalently”. In a seminal work, Gordon Allport [2] outlines the basic principles of categorization and argues that categorization is an important cognitive source of prejudice and stereotyping. Psychologists have developed Allport’s ideas by

¹Camerer [4] surveys recent developments in behavioral economics, and Rubinstein [22] discusses models of bounded rationality.

investigating how the categorization process operates (see e.g. Rosch [19] and Mervis and Rosch [17]), and how categorization affects social interactions and in particular how it leads to prejudice (see [6] for a survey). Recently, Fryer and Jackson [9] developed a model of how experiences are optimally sorted into categories and how categorization affects social decision making.

In the context of individual choice and choice from lists, the decision maker uses categories to effectively summarize relevant information. He evaluates the elements of the list in the order in which they appear. When encountering the next element in the list, the decision maker determines the next category as a function of the element just seen and all the relevant past information, which is summarized in the current category. When the decision maker decides to stop or the list ends, he chooses an element from the list as a function of all the information he has, which includes the current category and the last element he encountered.

The fewer categories a decision maker employs in making choices, the fewer cognitive resources he utilizes and hence the simpler his behavior is. The *procedural complexity* of a choice rule is the minimal number of categories required to implement the rule. This measure of complexity highlights the *fixed* cognitive cost associated with refining the information used in the decision making process, abstracting from search costs that highlight the *marginal* cognitive cost associated with searching and evaluating the next element in the list.

The paper investigates the connection between this basic form of procedural complexity and framing effects. In the context of choice from lists, there are two sources of framing effects. First, choices may depend on the order in which the alternatives appear. For example, a decision maker may tend to choose elements in the beginning or in the end of the list, thus displaying a primacy or a recency effect. Second, choices may depend on the number of times a given element appears in the list, such as a tendency to favor elements that appear multiple times in the list. I also consider framing effects that depend on a default alternative, such as the status-quo bias [12].

The benchmark behavior in the analysis is utility maximization, or rational choice. When maximizing a utility function u , a category includes all the lists that share the same u -maximal element, since future information processing is identical for all of them – the chosen element is either that same u -maximal element if all subsequent elements are u -inferior to that element, or the u -largest element among all subsequent elements. A category cannot, however, include two lists that have a different u -maximal element, because the decision maker chooses differently if the next and last element is the u -minimal element. Hence, the

procedural complexity of rational choice nearly equals the number of feasible alternatives.² Satisficing is much simpler, since the only relevant past information is whether a satisfactory element already appeared in the list.

The first main result of the paper is that utility maximization is procedurally simplest among all choice rules that are robust to framing effects.³ This result consists of three parts. First, rational choice rules are *uniquely* simplest among all choice rules that are order-independent, i.e, rules that choose the same element from every two lists that are permutations of one another. Second, rational choice is simplest among all choice rules that are order-independent for two-element lists, but possibly order-dependent for larger lists. Put another way, *any choice rule that is strictly simpler than rational choice is order-dependent for some two-element list*. Third, the first two parts carry over to framing effects that result from repetition of elements in the beginning and in the end of the list.

The second main result discusses situations in which the number of feasible alternatives is large compared to the information processing abilities of the decision maker. In such situations, utility maximization may be replaced with a simpler choice rule that is “close” to utility maximization. For example, in an organization, a manager may replace utility maximization with a simpler rule if a non-sophisticated subordinate is to implement that rule. I investigate this *choice design* problem under the assumption that the objective is to find a choice rule that minimizes the maximal utility loss, or regret, associated with making choices.

I establish that under mild restrictions the generically unique solution to this choice design problem is a history-dependent satisficing procedure. In this procedure, the decision maker sacrifices using several aspiration thresholds that adjust according to the identity of the elements that appeared so far in the list. Unlike in optimal search with no-recall or with perfect-recall, even when a history-dependent satisficer evaluates all the elements in the list, he may end up choosing an element that is neither the utility maximizing element (as in search with perfect-recall) nor the last element in the list (as in search with no-recall.)

²No category is needed for lists that include the u -maximal element, because no future information processing is required—the decision maker can stop and choose this element.

³Several previous papers have pointed out that rational choice is “simple”. Campbell [5] and Bandyopadhyay [3] show that intuitive procedural properties characterize choice correspondences that can be represented as the maximization of a binary relation. Rubinstein [21] suggests that the frequent appearance of order relations in natural language can partly be explained by the fact that order relations are easy to describe. Kalai [15] shows that rational behavior is easy to learn in the Probably-Approximately-Correct learning model.

This optimal history-dependent satisficing procedure displays several framing effects: (i) Primacy effect – moving an element, except maybe the last, toward the beginning of the list improves the likelihood it is chosen, (ii) Recency effect – moving an element that is not chosen to the last position in the list may make it the chosen element, (iii) Choice overload – adding elements to a list may result in a u -worse choice, and (iv) Default tendency – in the presence of a default, the decision maker ignores all the elements in some utility range above the default, except the last element in the list. Thus, particular kinds of framing effects, which are considered “biases” in some contexts, are actually optimal when taking procedural costs into account.⁴

2 Describing choice behavior

To describe choice behavior, I use the notion of a choice function from lists (see Rubinstein and Salant [23].) Let X be a finite set of N elements. A *list* is a finite sequence of elements from X . A given element may not appear, appear once, or appear multiple times in a given list. A *choice function from lists* assigns to every non-empty list an element from the list, interpreted as the chosen element.

Choice functions from lists can reflect a variety of framing effects that depend on the order of the alternatives. For example,

Example 1. Satisficing (Simon [25]). The decision maker has in mind a value function v over X and an aspiration threshold v^* . He chooses the first element x in the list with value $v(x) > v^*$ and, if there is no such element, he chooses the last element.

Thus, a satisficer displays a primacy effect (moving a satisfactory element toward the beginning of the list increases the likelihood it is chosen) and a recency effect (moving a non-satisfactory element to the last position in the list increases the likelihood it is chosen.)

The following two examples describe more subtle order effects.

Example 2. Maximizing with a bias. The decision maker has in mind a utility function u and a “bias” function b from X to R_+ . He evaluates the elements of a list in order. He begins by designating the first element as a “favorite”. When reaching the i 'th element, a_i , the decision maker replaces the current favorite y with a_i if $u(a_i) > u(y) + b(y)$, i.e., he gives a bonus to the current favorite. When the list ends, the decision maker chooses the current

⁴Wilson [27] makes a similar argument in the context of inference-making with limited memory.

favorite. The bonus $b(y)$ may be interpreted as a “mental” endowment effect [12].

Example 3. Contrast. The decision maker classifies the elements of X as “conventional” or “non-conventional”. He chooses the first conventional element that appears after two consecutive non-conventional elements. If there is no such element, the last element is chosen.

In addition to framing effects that depend on the order of the alternatives, choice functions from lists can also describe framing effects that depend on the number of times a given alternative appears in the list. The following is an extreme example.

Example 4. Choosing the most popular element. The decision maker chooses from every list the element that appears the largest number of times in the list. If there is more than one such element, the decision maker chooses the one among them that appears first on the list.

Standard frame-independent choice behavior in which neither the order of the alternatives nor repetition affect choice can also be described in the model. The following is a canonical example that will be the center of attention in the subsequent analysis.

Rational choice. The decision maker has in mind a strict preference relation (i.e., complete, asymmetric and transitive) \succ over X . He chooses the \succ -maximal element from every list.

3 The decision making process

A choice function from lists describes *what* the decision maker chooses. It misses, however, the procedural aspects of *how* he makes actual choices. Analyzing these aspects requires specifying the decision making process. While the process may vary across individuals and across choice problems, it seems that in many real-life situations information processing is *sequential* (i.e., the decision maker processes the elements of the list one after the other) and *categorization-based* (i.e., the decision maker classifies past information to one of several categories.)

To capture sequential decision-making with the aid of categories, I use the automaton model.⁵ I first describe the automaton model and demonstrate through examples how an

⁵The automaton model is one of the basic tools developed in computer science to investigate computational complexity (see Hopcroft and Ullman[10]). Rubinstein [20], Neyman [18], Abreu and Rubinstein [1] and others adapt the automaton model to study procedural aspects in repeated games. Dow [7] and Wilson [27] study procedural models of inference-making, which may be thought of as variations of the automaton

automaton makes choices. I then relate the automaton model to sequential information processing with the aid of categories, and introduce a measure of complexity that corresponds to the number of categories the decision maker uses in making choices.

3.1 Automaton

The building block of an automaton is a set of information states, which may be thought of as the categories the decision maker uses to process information. An automaton reads the elements of the list in order, and processes information by moving between its states according to the alternatives it encounters. When the automaton decides to stop or the list ends, the automaton outputs a chosen element.

Formally, the four components of an automaton are (i) a set Q of information states; (ii) an *initial* state $q_0 \in Q$ in which the automaton starts operating; (iii) a transition function $g : Q \times X \rightarrow Q \cup \{Stop\}$ – If the automaton is in state q and it encounters the element x , it moves to state $g(q, x)$ or Stops; and (iv) an output function $f : Q \times X \rightarrow X$ – If $g(q, x) = Stop$ or x is the last element in the list, the automaton produces the output $f(q, x)$.

An automaton operates as follows. For every list $L = (a_1, \dots, a_k)$, it starts in the initial state q_0 and reads the elements of the list in order. It processes information by moving between states: when the automaton is in state q (which is initially q_0) and it reads the element a_i , it moves to state $g(q, a_i)$ or stops. If the automaton decides to stop or a_i is the last element in the list, the automaton chooses the element $f(q, a_i)$.

Efficient implementation. An automaton *implements* a choice function from lists C if it outputs the chosen element $C(L)$ from every list L . An automaton implementing C is *efficient* if there exists no automaton with fewer states that implements C .

In what follows, I will analyze automata that efficiently implement choice functions from lists.

3.2 Examples

I use transition diagrams to demonstrate how an automaton operates. The circles in the diagram correspond to the states of the automaton. The left-most circle is the initial state.

model.

The Stop sign represents the ability of the automaton to Stop. Edges represent transitions: a character x on an edge from state q to state q' or to the Stop sign indicates that given that the automaton is in state q and it reads the character x , the automaton moves to state q' or stops. The mapping $x \rightarrow f(q, x)$ below a state q describes the output function in state q .

Let $X = \{1, 2, 3, 4\}$.

Satisficing. The one-state automaton in Figure 1 implements a satisficing procedure in which 3 and 4 are the satisfactory elements. Indeed, when the automaton is in state q_0 and

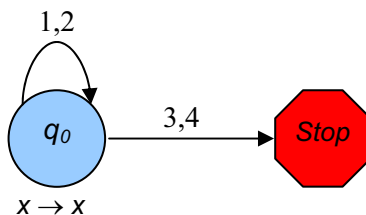


Figure 1: Satisficing

it encounters the element x , there are two possibilities. If $x \in \{3, 4\}$ or x is the last element in the list, then the automaton stops and chooses $f(q_0, x) = x$. Otherwise, the automaton stays in state q_0 , and reads the next element in the list. Thus, from the list $(4, 2, 1)$ the automaton chooses 4. From the list $(2, 1)$ it chooses 1 because 1 is the last element in the list and the automaton did not stop before seeing it.

Maximizing with a bias. Let $u(1) = 0$, and $u(x) = x$ otherwise. Set the bonus associated with each element at $b = 1.5$. The two-state automaton in Figure 2 implements a “maximizing with a bias” procedure based on these primitives.

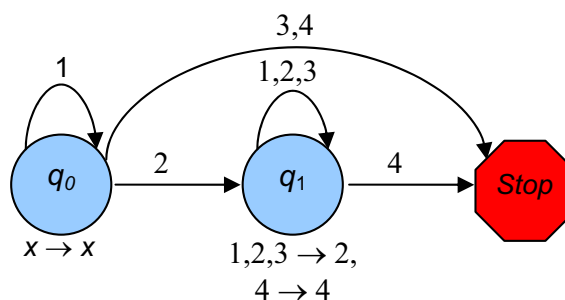


Figure 2: Maximizing with a bias

Contrast. The three-state automaton in Figure 3 implements a contrast procedure in which the elements 2 and 3 are “conventional”, and 1 and 4 are “non-conventional”.

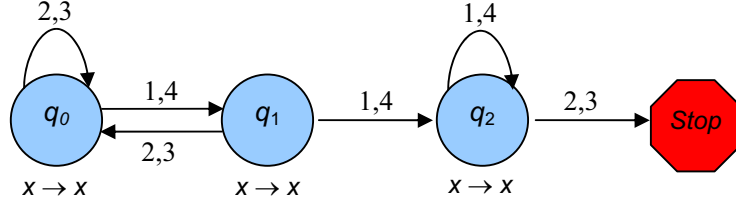


Figure 3: Non-conventional elements highlight the attractiveness of conventional ones

3.3 States and categorization

The states of an efficient automaton may be interpreted as categories the decision maker uses to process information, where a category contains all the lists that the decision maker processes in the same way.

To illustrate the connection between states and categories, consider the maximizing with a bias procedure from the previous subsection. In this procedure, the decision maker processes the element 3 differently depending on whether the element 2 appeared before it or not. If it did, the decision maker ignores 3 because $u(3) < u(2) + b(2)$. Otherwise, the decision maker chooses 3 because $u(3) + b(3)$ is larger than the utility of any other element. Thus, in terms of information processing, a decision maker that still did not make a choice distinguishes between two categories of lists: lists that include 2 and lists that do not include 2. These two categories correspond to states q_1 and q_0 in Figure 2.

Formally, for two lists L_1 and L_2 , let (L_1, L_2) be the concatenated list that consists of the elements of L_1 followed by the elements of L_2 . Given a choice function C , a list L is *C-undecided* if L is empty or if there is a continuation L' such that $C(L, L') \neq C(L)$. Otherwise, L is *C-decided*. Intuitively, a list is *C-undecided* if after seeing that list, the decision maker is still uncertain about his choice. For example, in satisficing a list is undecided if and only if it does not contain satisfactory elements. Two lists L_1 and L_2 are *C-equivalent* if $C(L_1, L) = C(L_2, L)$ for every non-empty list L . Otherwise, L_1 and L_2 are *C-separable*.

Intuitively, two lists are C -equivalent if after seeing one of them, future choices are the same as after seeing the other. In satisficing, every two undecided lists are equivalent because for every continuation, the first satisfactory element in the continuation is chosen.

Category. A *category* is an equivalence class of the binary relation \sim_C defined by $L_1 \sim_C L_2$ if L_1 and L_2 are C -undecided and C -equivalent lists.

Thus, a category contains all the lists for which future information processing is required (i.e., the lists are undecided) and if identical (i.e., the lists are equivalent.)

Informational index. The *informational index* of a choice function C is the number of equivalence classes, or categories, of the relation \sim_C .

For example, in the contrast procedure, a list is undecided if and only if it contains no three consecutive elements, such that the first two are non-conventional and the third one is conventional. Among undecided lists, the decision maker classifies lists to three categories: (i) lists ending with a conventional element, (ii) lists ending with a non-conventional element preceded by a conventional one, and (iii) the remaining lists. Thus, the informational index is three.

The following result establishes the connection between the number of states in an efficient automaton and the number of categories the decision maker uses to process information. The result is a modification to the context of individual decision making of the Myhill-Nerode Theorem [10] from the computer science literature and a related result by Kalai and Stanford [14] from the repeated games literature.

Theorem 1 *The informational index of C is identical to the number of states in an efficient automaton implementing C .*

The intuition for the result is as follows. Assume the number of categories used for information processing is K . Thinking about each category as a state, and linking two states q_1 and q_2 by a transition $g(q_1, a) = q_2$ if the category corresponding to state q_1 contains the list L and the category corresponding to q_2 contains the list (L, a) , we can construct an automaton implementing the function. This automaton's output $f(q, a)$ in state q when seeing the element a is defined by $C(L, a)$, where L is any list that belongs to the category corresponding to state q . Thus, the informational index is weakly larger than the number of states in an efficient automaton. If, however, the informational index were strictly larger than the number of states, then there are two lists from different categories, such that the

automaton reaches the same state after processing each of them. But then these two lists are equivalent because the automaton’s future actions depend only on its state and future information, contradicting the fact that the lists are from different categories. Thus, the informational index is equal to the number of states in an efficient automaton. The proof of Theorem 1 is left to the appendix.

3.4 Procedural complexity

Theorem 1 links categorization to the number of states in an efficient automaton, and motivates the definition of procedural complexity in the model.

Procedural complexity. The *procedural complexity* of a choice function from lists is the number of states of an efficient automaton implementing the function.

In addition to categories used for information processing, states may also be interpreted as representing “states of mind” of the decision maker. In satisficing, for example, the decision maker is either unsatisfied (in the initial state) or satisfied (in which case he stops and makes a choice), and in the contrast example, states represent how determined the decision maker is to make a conventional choice. As the decision maker sees more non-conventional elements, he becomes more convinced to make a conventional choice.

Of course, there are other complexity measures of interest. In the basic search model, for example, the source of procedural complexity is the “marginal” cost associated with searching and evaluating the next element in the list rather than the “fixed” cost of implementing the choice rule. In the context of repeated games, Lipman and Srivastava [16] measure complexity by the responsiveness of an automaton to changes in the history of the game, and Eliaz [8] studies the complexity of the transition function. Analyzing these and other complexity measures is beyond the scope of the current paper.

3.5 Simplest choice behavior

Consider an automaton M with one state q_0 that implements a choice function C . Then, $f(q_0, x)$ equals x , or else M does not choose correctly from some one-element list. Classifying an element x as “satisfactory” if $g(q_0, x) = Stop$, or “non-satisfactory” if $g(q_0, x) = q_0$, we have that C is consistent with choosing the first satisfactory element from every list, or the last element in the list if no satisfactory element exists. In addition, as demonstrated

in section 3.2, any satisficing procedure can be implemented using a one-state automaton. Thus,

Observation 1 *A choice function has minimal complexity if and only if it can be represented as a satisficing procedure.*

4 Rational choice and other behaviors

I now compare the procedural complexity of rational choice or, utility maximization, to that of other behaviors. After establishing that the complexity of utility maximization nearly equals the number of feasible alternatives, I demonstrate how situational cues simplify the difficulty of maximizing. I then prove that any choice function that is procedurally simpler than rational choice displays certain framing effects even for some two-element list.

The method of proof I use in this section is based on Theorem 1. To show that the complexity of a given choice function C is at least K , I identify a collection of K C -undecided and C -separable lists. To show that complexity is exactly K , I find a collection of K C -undecided and C -separable lists and show that any other list is either C -equivalent to one of these lists or C -decided.

4.1 The complexity of rational choice

Consider a rational decision maker who wishes to maximize the preference relation $4 \succ 3 \succ 2 \succ 1$. He can do so with three states, or categories, that correspond to the elements 1,2, and 3 as follows. In a state that corresponds to the element x , the decision maker stays in this state as long as he sees elements that are \succ -inferior to x ; when he sees an element y that is \succ -superior to x , he moves to the state corresponding to y . When seeing the element 4, the decision maker stops and chooses it. Figure 4 illustrates this construction. In the figure, state q_i corresponds to how information is processed conditional on the element $i + 1$ being the \succ -maximal element seen so far (the figure excludes self-loops.) This four-element example extends to any number of alternatives as follows.

Observation 2 *The procedural complexity of rational choice is $N-1$, where $N = |X|$ is the number of feasible alternatives.*

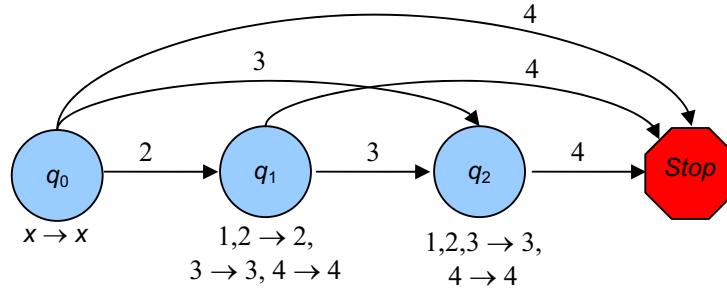


Figure 4: Rational choice

Proof. Let C be a rational choice function that maximizes the preference relation \succ , x_{\max} be the \succ -maximal element in X , and x_{\min} be the \succ -minimal element in X . By Theorem 1, it suffices to show that the informational index of C is $N - 1$. Consider the collection of one-element lists $\{(x) \mid x \in X \setminus \{x_{\max}\}\}$. These lists are C -undecided and pairwise C -separable (consider the continuation (x_{\max})). Thus, the index of C is at least $N - 1$.

Consider any other C -undecided list L . If L is empty then it is C -equivalent to (x_{\min}) . If L is C -undecided and non-empty, then it does not contain x_{\max} , and in particular $C(L) \in X \setminus \{x_{\max}\}$. Moreover, L is C -equivalent to the one-element list $(C(L))$, because for every continuation the \succ -maximal element among $C(L)$ and the \succ -best element in the continuation is chosen. Thus, there is no larger collection of C -undecided lists that are pairwise C -separable, implying that the index is $N - 1$. ■

It is straightforward to construct an automaton with $N - 1$ states that implements rational choice following the intuition of Figure 4. Indeed, let x_{\max} be \succ -maximal in X and x_{\min} be \succ -minimal in X . Denote the set of states by $Q = X \setminus \{x_{\max}\}$ and the initial state by $q_0 = x_{\min}$, so that every state is associated with a different element in $X \setminus \{x_{\max}\}$. When encountering an element $x \neq x_{\max}$ in state q , move to the state that corresponds to the \succ -better element among q and x . When encountering $x = x_{\max}$, stop. Finally, output the \succ -better element among q and x .

4.2 Rational choice and situational cues

Situational cues may reduce the burden of maximization as the following examples demonstrate.

1. Restricted domain of alternatives. The decision maker is sometimes able to partially affect the identity of the alternatives that appear in choice problems. For example, many websites enable consumers to choose products only in a certain price range. In such cases, the domain of feasible alternatives shrinks from X to $\bar{X} \subset X$, so a rational decision maker has to entertain fewer categories and maximization becomes easier.

2. Restricted domain of choice problems. The decision maker may also be able to affect the ordering of the alternatives. For example, when making an online purchase, the decision maker can often sort the items according to specific attributes, such as price or popularity. The resulting ordering may be correlated with the decision maker's preferences, and may therefore simplify maximization. For example, if the decision maker's preference relation is single-peaked with respect to price, and listing according to price is possible, then maximizing becomes simpler: once the decision maker "passes his peak" he chooses the \succ -maximal alternative between the \succ -best alternative before the peak and the first one after the peak. That is, no categories are required for processing elements that appear after the peak. Of course, if pre-sorting generates an ordering that is identical to the individual's preference relation, making choices becomes trivial.

3. Default alternative. Assume the decision maker is endowed with a default alternative δ . When choosing from a list, the decision maker either chooses an element from the list or keeps the default. In order to incorporate the default into the automaton model, the range of the output function f should be modified from X to $X \cup \{\delta\}$. Rational choice in the presence of a default is to choose from every list the \succ -maximal element x if $x \succ \delta$, and δ otherwise. As long as there are elements in X that are \succ -inferior to δ , the complexity of this procedure is smaller than that of rational choice without a default, because all the lists that include elements, which are not \succ -superior to the default, can be pooled in the same category. Moreover, if the default alternative benefits from a utility bonus in addition to its intrinsic value (as in the status-quo bias [24]), then the set of elements that are superior to δ may shrink further, and the resulting choice function is even simpler.

In addition to situational cues, internal mechanisms may also simplify maximization. In the maximizing with a bias example, the decision maker gives a "mental" bonus $b(x)$ to the favorite x in addition to its utility $u(x)$. In this case, an element x for which $u(x)+b(x) \geq u(y)$ for every other element y , does not "require" a state. Indeed, once x is seen, it can either be chosen, if its utility is higher than the utility plus the bonus of the current favorite, or

ignored in any other case. Thus, if bonuses are high enough, maximizing with a bias requires fewer categories than rational choice.⁶

4.3 Rational choice and framing effects

Rational choice is frame-independent, i.e., it is not affected by the ordering of the alternatives and by whether an element appears multiple times in a list. I now show that rational choice is procedurally simplest among all choice functions that are order-independent and among all choice functions that are repetition-independent. Put another way, any choice function that is simpler than rational choice displays framing effects that depend on the ordering of the alternatives and on repetition. I start with order effects.

Order-independence for pairs. A choice function C is *order-independent for pairs* if for every two elements $a, b \in X$, $C(a, b) = C(b, a)$.

Observation 3 *Any choice function that is order-independent for pairs is at least as complicated as rational choice.*

Proof. Let C be order-independent for pairs. Denote by $X^* \subseteq X$ the set of all elements $x \in X$ for which there exist elements $w(x), l(x) \in X \setminus \{x\}$ such that $C(x, w(x)) = w(x)$ and $C(x, l(x)) = x$. The cardinality of the set X^* is $\geq N - 2$. Otherwise, there are two elements x_1 and x_2 that are both “winners” ($C(x_i, x) = x_i$ for every $x \in X$) or both “losers” ($C(x_i, x) = x$ for every $x \in X$), which is impossible. Indeed, if x_1 and x_2 are winners, then $C(x_1, x_2) = x_1$ and $C(x_2, x_1) = x_2$, in contradiction to order-independence for pairs. Similarly, there cannot be two losers.

For every $x \in X^*$, the one-element list (x) is undecided since $C(x, w(x)) = w(x) \neq C(x)$. The list (x) is separable from the empty list because $C(x, l(x)) = x \neq C(l(x))$. For every two distinct elements $x, y \in X^*$, the lists (x) and (y) are separable because $C(x, l(x)) = x \neq C(y, l(x))$. Thus, the collection of one-element lists (x) , where $x \in X^*$, and the empty list are undecided and separable. Therefore, the complexity of C is at least $N - 1$. ■

If a choice function is order-independent for all lists, a stronger result can be obtained. Theorem 2 shows that if a choice function is order-independent for *every* list and cannot

⁶Exceptions include cases in which for every element x there is $y \in X$ such that $u(x) + b(x) < u(y)$, and in addition $u(x_{\min}) + b(x_{\min}) > u(z)$ where x_{\min} is the u -minimal element in X and $z \in X \setminus \{x_{\min}\}$.

be represented as the maximization of a strict preference relation, then it is *strictly* more complicated than rational choice. For example, the behavior of a decision maker that chooses from every list the median-priced item is order-independent and cannot be represented by the maximization of a preference relation. Hence, it is more complicated than rational choice.

Order-independence. A choice function C is *order-independent* if for every list (a_1, a_2, \dots, a_k) and every permutation σ of $\{1, \dots, k\}$,

$$C(a_1, a_2, \dots, a_k) = C(a_{\sigma(1)}, a_{\sigma(2)}, \dots, a_{\sigma(k)}).$$

A choice function C is *rationalizable* if there exists a strict preference relation \succ such that for every list L , $C(L)$ is the \succ -maximal element in L . Otherwise, C is non-rationalizable.

Theorem 2 *Any non-rationalizable and order-independent choice function is strictly more complicated than rational choice.*

Proof. Let C be order-independent. By observation 3, the procedural complexity of C is at least $N - 1$. Assume it is exactly $N - 1$. I first state several implications of order independence and having complexity of $N - 1$. I then use these implications to prove that if C is not rationalizable, then complexity is at least N .

Fact 1. There is an element w such that $C(w, x) = w$ for every $x \in X$. Similarly, there is an element l such that $C(l, x) = x$ for every $x \in X$.

Proof. I prove the first part. The proof of the second part is analogous. Assume to the contrary that for every element x there exists an element $w(x) \neq x$ that “beats” x , i.e., $C(x, w(x)) = w(x)$. Because of order-independence, there is at most one element y such that $C(y, z) = z$ for every $z \in X$. Hence, for every element $x \in X$ except at most one, there exists an element $l(x)$ such that $C(x, l(x)) = x$. Therefore, there is a set $X^* \subseteq X$ of cardinality at least $N - 1$ such that if $x \in X^*$, then there are elements $w(x), l(x) \in X \setminus \{x\}$ satisfying $C(x, w(x)) = w(x)$ and $C(x, l(x)) = x$. Applying the second part of the proof of observation 3 to this case, complexity is at least N . ■

Let $Y = \{(x) \mid x \in X \setminus \{w\}\}$. By fact 1 and order-independence, Y is a collection undecided and pairwise separable lists. Because complexity is $N - 1$, every other list is either decided or equivalent to some $(x) \in Y$. This implies:

Fact 2. $C(L) = C(l, L)$ for every list L .

Proof. The empty list is undecided by definition, and hence equivalent to some $(x) \in Y$. Therefore, $l = C(l) = C(x, l) = x$ and thus $x = l$. That is, the empty list is equivalent to (l) implying that $C(L) = C(l, L)$ for every list L . ■

Fact 3. $C(L) = w$ if w is an element of L .

Proof. Otherwise, by order-independence the one-element list (w) is undecided, and hence equivalent to some $(x) \in Y$. But then $w = C(w, l) = C(x, l) = x$ in contradiction to the fact that $x \neq w$. ■

Fact 4. $C(a, L) = C(L)$ for every list L and for every element a that appears in L .

Proof. Assume to the contrary there is an element a and a list L that includes a such that $C(a, L) \neq C(L)$. By facts 2 and 3, $a \notin \{w, l\}$. Consider the list (a, a) . It is undecided because $C(a, a, w) = w$, and hence it is equivalent to some $(x) \in Y$. Thus, $x = C(x, l) = C(a, a, l) = a$ where the last equality is derived from fact 2 and order independence. Hence, (a, a) is equivalent to (a) . Denote by L' a sublist of L in which one instance of a is omitted. Because C is order-independent, $C(L) = C(a, L')$ and $C(a, L) = C(a, a, L')$. Because (a) and (a, a) are equivalent, $C(a, L') = C(a, a, L')$ implying that $C(L) = C(a, L)$ which is a contradiction. ■

To conclude the proof of the theorem, I now show that if C is not rationalizable, complexity is at least N . Denote by $S(L)$ the set of distinct elements that appear in the list L . By fact 4 and order-independence, C must satisfy $C(L) = C(L')$ if $S(L) = S(L')$. Thus, without loss of generality, we can assume that C is defined over sets.

Hence, if C is not rationalizable, then it violates the standard Independence of Irrelevant Alternatives property. That is, there are sets $A \subset B$ such that $C(L_B) \in A$ but $C(L_A) \neq C(L_B)$, where L_S is some listing of the elements of the set S . The list L_A is undecided because $C(L_A) \neq C(L_A, L_{B \setminus A})$, and hence equivalent to some $(x) \in Y$. In particular, $C(x, L_{B \setminus A}) = C(L_A, L_{B \setminus A}) = C(L_B)$, and thus because $C(L_B) \notin B \setminus A$ we obtain that $x = C(L_B)$. Thus, L_A is equivalent to $C(L_B)$, and in particular, $C(L_A, l) = C(C(L_B), l)$. By fact 2 and order-independence, $C(L_A, l) = C(L_A)$ and $C(C(L_B), l) = C(L_B)$ and thus $C(L_A) = C(L_B)$ in contradiction to $C(L_A) \neq C(L_B)$. ■

Similar results can be obtained for choice functions that are robust to the repetition of elements in the beginning and the end of a list.

Repetition-independence. A choice function C is *repetition-independent* if $C(L) = C(L, x) = C(x, L)$ for every list L and for every element x that appears in L . If this condition holds

only for two-element lists, then C is *repetition-independent for pairs*.

The following observation is an immediate implication of observation 3 and theorem 2.

Observation 4 *Any choice function C that is repetition-independent for pairs is at least as complicated as rational choice. If C is repetition-independent and non-rationalizable, then it is strictly more complicated than rational choice.*

Indeed, assume C is repetition-independent for pairs. Then, for every two elements a and b , $C(a, b) = C(a, b, a) = C(b, a)$. Thus, C is order-independent for pairs, and the first part of observation 4 follows. Similarly, if C is repetition-independent then it must be order-independent. Otherwise, if C is order-dependent then there exist lists L_1 and L_2 that are permutations of one another such that $C(L_1) \neq C(L_2)$. By repetition independence, however, $C(L_1) = C(L_1, L_2) = C(L_2)$, which is a contradiction.

Observation 5 summarizes the relationship between rational choice and simpler behaviors.

Observation 5 *Any choice function C that is strictly simpler than rational choice displays order-dependence and repetition-dependence for pairs. If C is non-rationalizable and weakly simpler than rational choice, it displays some form of order-dependence and repetition-dependence.*

5 Choice design

In rational choice, the number of categories used for information processing nearly equals the cardinality of the outcome space. Hence, when the outcome space is large, rational choice is complicated. This motivates designing choice procedures that are “close” to maximizing utility but require fewer categories.

The first step in approaching this problem is to determine how close a given choice rule is to utility maximization. If the choice designer knows the details of the random process that generates lists, it seems natural to use a measure of “closeness” that depends on the specific process such as the expected distance of the choice rule from maximizing utility. If, however, the designer is uncertain about the actual process that generates lists or, alternatively, seeks a choice rule that performs well with respect to multiple processes, it seems more natural to have a process-independent measure such as minimizing the maximal distance from utility

maximization. I will focus on latter criterion, and comment on the usage of the former when stating the main theorem of this section.

Let u be a utility function and C an arbitrary choice function. Given a list L , let $u(L)$ be the utility of the u -maximal element in L , and let $u(C(L))$ be the utility of the element that C assigns to L . The regret of C with respect to u , $regret_u(C)$, is the maximal utility loss associated with making choices according to C rather than maximizing u :

$$regret_u(C) = \max_{L \in \mathcal{L}} \{u(L) - u(C(L))\},$$

where \mathcal{L} is the set of all non-empty lists.

Using $regret_u(C)$ as a measure of closeness between u and C , the objective of the following choice design problem is to minimize regret subject to having limited abilities to process information:

$$\text{MinReg}(K) : \min_{C : \text{comp}(C)=K} regret_u(C)$$

where $\text{comp}(C)$ is the procedural complexity of C . Clearly, $\text{MinReg}(K)$ has a solution because the number of choice functions with complexity K is finite.

To characterize the choice functions that solve $\text{MinReg}(1)$, let x_{\max} be the u -maximal element in X , and let u_{\max} be the utility of x_{\max} . Define x_{\min} and u_{\min} similarly.

Observation 6 *The generically unique choice function that solves $\text{MinReg}(1)$ is a satisficing procedure with aspiration threshold $t_0 = \frac{u_{\min} + u_{\max}}{2}$.*⁷

Proof. Let y be the u -smallest element such that $u_y \geq t_0$ and z the u -largest element such that $u_z \leq t_0$. Then the regret of satisficing with threshold t_0 is $V = \max \{u_{\max} - u_y, u_z - u_{\min}\}$. To prove Observation 6, it is enough to show that any automaton implementing a choice function C that solves $\text{MinReg}(1)$ operates as follows:

- (i) It stops when it encounters an element x with $u(x) > t_0$, i.e., $g(q_0, x) = \text{Stop}$;
- (ii) It stays in the initial state when it encounters an element x with $u(x) < t_0$, i.e., $g(q_0, x) = q_0$.

I prove (i). The proof of (ii) is analogous. Assume to the contrary that there exists an element x such that $u_x > t_0$ yet $g(q_0, x) = q_0$. Then the regret of C is at least $u_x - u_{\min}$,

⁷If there exists an element x such that $u(x) = t_0$, there is an additional choice rule that solves $\text{MinReg}(1)$, which is identical to the above satisficing procedure except for choosing x when x appears rather than ignoring it.

e.g., for the list (x, x_{\min}) . Because $u_x > u_z$, we obtain that $u_x - u_{\min} > u_z - u_{\min}$. Because $u_x \geq u_y \geq t_0$, we obtain that $u_x - u_{\min} \geq u_y - u_{\min} \geq u_{\max} - u_y$, where at least one of the two inequalities is strict because either $u_x > u_y$ or $u_y > t_0$. Thus, the regret of C is larger than V . ■

For $K > 1$, multiple choice functions may solve $\text{MinReg}(K)$. I now demonstrate through an example the problems associated with some of these choice functions, and present corresponding refinements.

Let $X = \{x_0, x_1, x_2, x_3\}$ with $u(x_1) = 1.5$ and $u(x_i) = i$ otherwise. There are multiple choice functions that solve $\text{MinReg}(2)$, they all have a regret of 1, and each of them follows the procedural skeleton of Figure 5.⁸

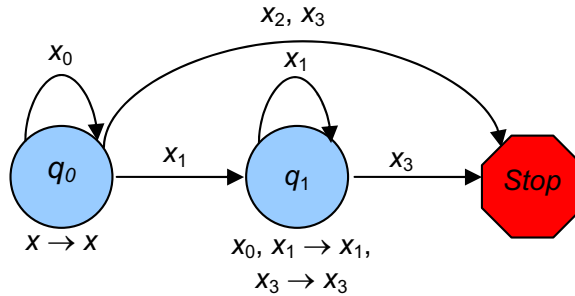


Figure 5: $\text{MinReg}(2)$

Figure 5 does not specify the element outputted is state q_1 when seeing x_2 , $f(q_1, x_2)$, and the corresponding transition $g(q_1, x_2)$. These vary across automata that solve $\text{MinReg}(2)$.

First, the element $f(q_1, x_2)$ may equal x_1 even though $u(x_2) > u(x_1)$. To see this, note that if $g(q_1, x_2) = q_1$ then $f(q_1, x_2)$ is the chosen element only in lists that contain x_1 , do not contain x_3 , and have x_2 as the last element. The regret associated with any such list is 0.5. While $f(q_1, x_2) = x_1$ is a feasible output, changing $f(q_1, x_2)$ to be x_2 strictly improves the performance of the automaton for some lists while not affecting performance for the remaining lists. This is clearly a desirable property:

u-Efficiency. A choice function C satisfies u -Efficiency if there exists no other choice function C' with the same complexity such that $u(C'(L)) \geq u(C(L))$ for every list L , and $u(C'(L')) > u(C(L'))$ for at least one list L' .

Second, $g(q_1, x_2)$ may be either q_1 or $Stop$. If the decision maker stops when seeing x_2 in state q_1 , then conditional on seeing the one-element list (x_1) , the regret associated with

⁸These statements are immediate implications of Claims 1 and 2 in the appendix.

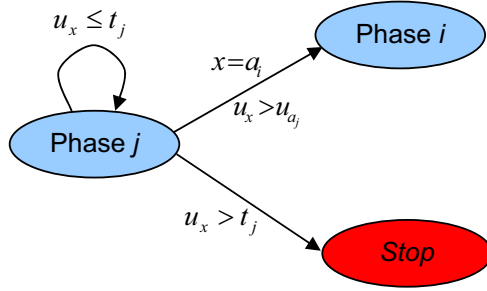


Figure 6: History-dependent satisficing

all possible continuations of (x_1) , denoted by $\text{regret}_u(C \mid (x_1))$, is at least 1 (consider e.g. the continuation (x_2, x_3)). If, however, the decision maker stays in state q_1 when seeing x_2 then $\text{regret}_u(C \mid (x_1))$ is at most 0.5. Minimizing this “conditional” regret is the second refinement I use. Formally, let $\text{regret}_u(C \mid L) = \max_{L' \in \mathcal{L}} \{u(L, L') - u(C(L, L'))\}$.

regret-Efficiency. A choice function C satisfies *regret-Efficiency* if there exists no other choice function C' with the same complexity such that $\text{regret}_u(C' \mid L) \leq \text{regret}_u(C \mid L)$ for every list L that is C - and C' -undecided, with at least one strict inequality.

I now describe the unique choice function that solves $\text{MinReg}(K)$ and satisfies u - and *regret-Efficiency*.

History-dependent satisficing. A *history-dependent satisficing* procedure with K phases is characterized by K aspiration elements a_0, \dots, a_{K-1} and K corresponding thresholds t_0, \dots, t_{K-1} . For every list, the procedure starts in phase 0. In phase j , $0 \leq j \leq K - 1$, the next element x in the list is processed as follows (see Figure 6 for an illustration):

- (i) If x is the last element in the list, choose the u -maximal element among a_j and x .
- (ii) if $x = a_i$ and $u_x > u_{a_j}$, switch to phase i . Otherwise,
- (iii) satisfice with the threshold t_j : if $u_x > t_j$ choose x and stop, and if $u_x \leq t_j$ ignore x and stay in phase j .

Thus, in a history-dependent satisficing procedure with K phases, there are K elements according to which the decision maker updates his aspiration threshold. The decision maker satisfices with respect to the current aspiration threshold. The following result is proved in the appendix.

Theorem 3 *There is a generically unique choice function that solves $\text{MinReg}(K)$ subject to u - and regret-Efficiency. This function is a history dependent satisficing procedure with K*

phases. In phase 0, $a_0 = x_{\min}$ and $t_0 = \frac{u_{\min} + u_{\max}}{2}$. In phase $j > 0$, a_j is the u -closest element to t_0 among the elements in the set $X \setminus \{a_0, \dots, a_{j-1}\}$, i.e., $a_j = \arg \min_{x \in X \setminus \{a_0, \dots, a_{j-1}\}} \{|t_0 - u(x)|\}$, and $t_j = \frac{u(a_j) + u_{\max}}{2}$.^{9 10}

Thus, the unique automaton that solves $\text{MinReg}(K)$ subject to the refinements classifies an alternative x as either satisfactory, “maybe” or non-satisfactory. If x is non-satisfactory, the automaton ignores it. If x is satisfactory, the automaton stops and chooses it. If x is a “maybe” element that is better than all the maybe elements seen so far, the automaton updates the aspiration level up to $\frac{u(x) + u_{\max}}{2}$, and continues processing information.

For example, let $X = \{x_1, \dots, x_{10}\}$ with $u(x_i) = i + \frac{1}{i}$, and consider solving the regret minimization problem with three categories subject to the refinements. By theorem 3, the initial threshold is the average between the utility of x_1 and x_{10} , which is $t_0 = 6\frac{1}{20}$. The two “maybe” elements are the u -closest elements to t_0 , x_5 and x_6 . When the decision maker sees an element with utility larger than x_6 in the initial phase, he stops and chooses it. When he sees an element with utility less than x_5 , he ignores it. When seeing x_5 , the decision maker updates the aspiration threshold to the average between the utility of x_5 and x_{10} , and continues satisficing with the new aspiration level. Similarly, when seeing x_6 , the decision maker updates the aspiration threshold to the average between the utility of x_6 and x_{10} . This history-dependent satisficing procedure is depicted in Figure 7.

5.1 Properties of the solution

The unique choice function C that solves $\text{MinReg}(K)$ subject to u - and *regret*-Efficiency displays several framing effects:

Primacy effect. Because aspiration thresholds increase along the list and because C chooses the first element above the current threshold, there is a primacy effect. That is,

⁹Theorem 3 extends to expected regret minimization, or alternatively expected utility maximization, as follows. Let P be a probability measure over X . Assume lists are generated by a stationary process: The first element in the list is drawn according to P . With probability μ the list ends. With probability $1 - \mu$, an additional element is drawn according to P . The process continues in the fashion until it stops. The analogue expected utility maximization problem subject to cognitive constraints is given by $(EU) \max_{C : \text{comp}(C)=k} E(u(C(L)))$. Then, the generically unique choice function that solves (EU) is a history dependent satisficing procedure with K phases.

¹⁰There are two non-generic cases. First, if there is an element x such that $u(x) = t_j$ for some j , a solution can output x in phase j rather than ignoring it. Second, if there are two solutions to the problem $\arg \min_{x \in X \setminus \{a_1, \dots, a_{K-2}\}} \{|t_0 - u(x)|\}$, then a_{K-1} can be either of them.

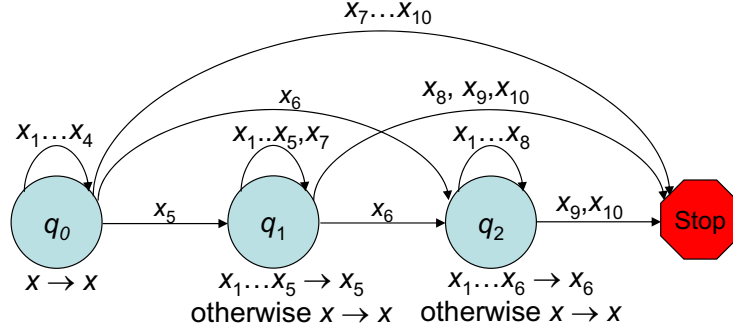


Figure 7: Optimal framing effects

moving any element (except maybe the last element in the list) toward the beginning of the list improves the likelihood that this element is chosen. For example, in figure 7, after seeing the “maybe” element x_6 the decision maker raises the aspiration level and thus ignores x_8 . Thus, he chooses x_6 from the list (x_6, x_8, x_1) . Switching the positions of x_6 and x_8 changes the chosen element to x_8 .

Recency effect. Because the decision maker can condition his choice on the last element he processes, there is a mild *recency* effect. That is, moving an element x that is not chosen to the last position in a list sometimes turns it to the chosen element. For example, in figure 7, the decision maker chooses x_6 from the list (x_6, x_8, x_1) , yet he chooses x_8 from the list (x_6, x_1, x_8) .

More choice is not always better. Adding an element to a list may result in an inferior choice because the additional element may increase aspiration level enough so that a previously chosen alternative is no longer chosen. This effect may be interpreted as an example of the choice overload hypothesis (see Iyengar and Lepper [11].) For example, in figure 7 the element x_8 is chosen from the list (x_8, x_1) yet x_6 is chosen from (x_6, x_8, x_1) .

Default tendency. When the decision maker is endowed with a default alternative δ with utility u_δ , he displays a *default tendency*: any element in some utility range above u_δ is ignored by the decision maker unless it is the last element in the list. To see this, let \bar{X} be the collection of elements in X that are u -superior to δ , and let $K < |\bar{X}|$. Solving $\text{MinReg}(K)$ over the space $X \cup \{\delta\}$ yields a history-dependent satisficing procedure in which $a_0 = \delta$ and $t_0 = \frac{u_\delta + u_{\max}}{2}$. In particular, the decision maker ignores all the elements in the utility range $[u_\delta, t_0]$, which are not aspiration elements, except maybe the last among them.

5.2 Costly categories

In $\text{MinReg}(K)$, the number of categories used for processing information is determined exogenously. It is straightforward to extend the analysis to a situation in which an explicit cost c is associated with each category:

$$(\text{MinReg}) \quad \min_C \{ \text{regret}_u(C) + c \cdot \text{comp}(C) \} \quad \text{s.t. } u\text{-Efficiency and } \text{regret}\text{-Efficiency}$$

In this case, the utility loss associated with a choice function C comes from two sources: the procedural cost corresponding to the complexity of C , and the cost associated with regret. Additional categories impose a procedural cost but may reduce regret.

Only choice functions with complexity $\leq N - 1$ are candidates to solve MinReg . Indeed, $\text{regret}_u(\cdot)$ is bounded below by zero and it is possible to reduce regret to zero by a rational choice function with complexity $N - 1$. Any additional category increases procedural costs but cannot reduce regret further. Moreover, by Theorem 3, for any fixed $1 \leq K \leq N - 1$, the unique choice function that solves $\text{MinReg}(K)$ subject to u - and regret -Efficiency is a K -phase history-dependent satisficing procedure. Thus, there are $N - 1$ candidate choice functions to solve MinReg . Generically, there is a unique K that solves this problem, and hence a unique history-dependent satisficing procedure that solves MinReg .

For example, let $X = \{x_1, x_2, \dots, x_N\}$ and assume $u(x_i) = i$. For simplicity, assume that N is an even number. Then, the unique solution to MinReg (even without the refinements) is full utility maximization if $c < 1/2$, and satisficing with the threshold $t_0 = \frac{u_{\min} + u_{\max}}{2}$ if $c > 1/2$. To see this, note that the regret associated with a one-state optimal automaton is $N/2 - 1$. Adding another state does not reduce regret. Adding two states reduces regret by one unit. More generally, adding a state to an automaton with an odd number of states does not reduce regret, while adding two states reduces regret by one unit. Therefore, if it is profitable to enlarge the number of states from one to three, it is also profitable to continue doing so until the automaton has $N - 1$ states.

Endogenizing the number of states generates predictions about the relation between procedural sophistication and the time it takes to make choices. Let $t_M(L)$ be the number of elements the automaton M reads from the list L before stopping. If processing each element takes one unit of time, then $t_M(L)$ measures the time until the decision maker makes a choice from L .

Different procedural costs. Assume two decision makers share the same utility function u but differ on procedural costs. Let c_i be the cost per-state of decision maker i .

Observation 7 *Let $c_1 \geq c_2$ and let M_i be the unique automaton that solves MinReg when the cost per-state is c_i . Then, $t_{M_1}(L) \leq t_{M_2}(L)$ for every list L .*

Proof. Denote by b_i the marginal benefit from moving from an optimal automaton with $i - 1$ states (that corresponds to the elements a_0, \dots, a_{i-1}) to an optimal automaton with i states. Then, the unique automaton that solves MinReg is an automaton of size k if and only if $f(m) = V_1 - \sum_{j=2}^m b_j + mc_i$ is minimized at k , where V_1 is the value of the solution to MinReg(1). If costs reduce from c_1 to c_2 , then the only change in $f(m)$ is that costs decrease by $m(c_1 - c_2)$, which is increasing in m . Thus, the optimal automaton under c_2 would have weakly more states and would nest (in the graph theoretic sense) the optimal automaton under c_1 . Hence, for every beginning L , either M_1 and M_2 reach the same state, or M_1 stops while M_2 does not, or M_2 reaches a state with a higher threshold than that of M_1 . In any case, the time until making a choice is larger in M_2 . ■

While the time for making choices increases as procedural costs decrease, the resulting choices are not always u -better. Indeed, reduced costs imply additional aspiration elements and possibly higher thresholds. Higher thresholds may result in ignoring elements that were history-dependent satisfactory before and turned non-satisfactory.

6 Concluding remarks

This paper investigated one procedural aspect of decision making. I considered automata implementing choice rules, and measured the procedural complexity of a given choice rule by the minimal number of states required for implementing it. The number of states captures the amount of information processing required for implementation.

In rational choice, information processing depends on the identity of the best element considered so far, so the complexity of rational choice nearly equals the number of feasible alternatives. Hence, any situational cue that makes some of the alternatives “irrelevant” such as a default alternative, simplifies rational choice.

When the outcome space is large, rational choice is complicated, and hence a decision maker may approximate rational choice in order to economize on procedural costs. As demonstrated in Section 4, any choice rule that uses less procedural resources than rational choice is affected by the ordering of the alternatives and by repetition even for two-element choice problems. Furthermore, theorem 3 establishes that an optimal tradeoff between maximizing utility and

minimizing procedural complexity results in more concrete framing effects: Primacy and recency effects, choice overload, and default tendency emerge as optimal when a “rational” decision maker economizes on procedural costs. Exploring whether additional behavioral phenomena, usually referred to as biases, emerge as natural solutions to decision problems that take procedural costs into account is a task for future research.

7 Appendix

7.1 Proof of Theorem 1

I prove Theorem 1 for choice functions with a finite informational index. If a choice function has an infinite informational index, then the number of states in any automaton implementing the function is infinite, and vice versa. The proof of the infinite case is similar to the proof of the finite case and is left to the reader.

Let C be a choice function with informational index $K < \infty$. I first show that any automaton implementing C has $\geq K$ states. I then construct an automaton implementing C with K states.

To see that any automaton implementing C has $\geq K$ states, assume to the contrary that there exists an automaton M with fewer than K states that implements C . The informational index of C is K and hence there exists a collection of K lists that are C -undecided and C -separable. Because the lists are C -undecided, M does not stop while or after processing any of them. Because there are K C -separable lists and fewer than K states, there are two C -separable lists L_1 and L_2 , such that M reaches the same state after processing any of them. But then M chooses the same element from (L_1, L) and (L_2, L) for every continuation L , in contradiction to L_1 and L_2 being C -separable.

I now construct an automaton M with K states that implements C . The states of M correspond to the K equivalence classes of the binary relation \sim_C . The initial state is the equivalence class of the empty list. Denote the equivalence class of a C -undecided list L by $[L]_{\sim_C}$. The transition function maps a state q and an element $a \in X$ to another state as follows: take any L in the equivalence class q and move to state $[(L, a)]_{\sim_C}$ if the concatenated list (L, a) is undecided; if (L, a) is decided, stop. Let $f(q, a) = C(L, a)$ for some list L in q .

The functions f and g are well-defined functions. Indeed, assume $L_1 \sim_C L_2$. Then $C(L_1, a) = C(L_2, a)$ for every element a and thus f is well-defined. In addition, because

L_1 and L_2 are undecided and $C(L_1, L) = C(L_2, L)$ for every non-empty list L , the list (L_1, a) is undecided if and only if (L_2, a) is undecided. If both are undecided, then $(L_1, a) \sim_C (L_2, a)$. Thus g is well-defined.

It remains to show that M implements C . I do so by induction on the length of the list. For every one-element list $L = (a)$, the automaton outputs $f(q_0, (a)) = C(L, a)$ for every list L in the equivalence class q_0 . Taking L to be the empty list we obtain that M outputs $C(a)$ as required. Consider a list $L = (L', a)$ where L' is non-empty. If L' is undecided, then by construction M reaches state $[L']_{\sim_C}$ after reading L' . Since a is the last element in the list, M outputs $f([L']_{\sim_C}, a)$. By the definition of f , $f([L']_{\sim_C}, a) = C(L', a) = C(L)$. If L' is decided, denote by L'' the shortest beginning of L' that is decided. By construction, the element M chooses from L is the element M chooses from L'' . By the induction assumption, the element M chooses from L'' is $C(L'')$, and by the fact that L'' is decided $C(L'') = C(L)$. Thus, M outputs $C(L)$ as required. ■

7.2 Proof of Theorem 3

As shown in observation 6 in the main text, the unique choice rule that solves $\text{MinReg}(1)$ is a history-dependent satisficing procedure with one phase. It is straightforward to verify u - and *regret*-Efficiency.

Consider the problem $\text{MinReg}(K)$ for $K \geq 2$. Assume that

- (1) For every element x and for every i , $u(x) \neq t_i$, and
- (2) there is a unique solution to $\arg \min_{x \in X \setminus \{a_0, a_1, \dots, a_{K-2}\}} \{|t_0 - u(x)|\}$.

Footnote 11 discusses the additional solutions when these properties do not hold.

I first identify the value of the solution to $\text{MinReg}(K)$ in Claim 1. Claim 2 identifies properties of any automaton that solves $\text{MinReg}(K)$. Claim 3 establishes that the history-dependent satisficing procedure of Theorem 3 solves $\text{MinReg}(K)$ subject to u - and *regret*-efficiency. Claim 4 establishes uniqueness and thus concludes the proof.

Relabel the elements a_1, \dots, a_{K-1} according to utility so that $u(a_1) > \dots > u(a_{K-1})$, and define $u_i = u(a_i)$. Let y be the u -smallest element such that $u_y > u_1$ and z the u -largest element such that $u_z < u_{K-1}$. Let $u_{\max} = \max\{u(x) \mid x \in X\}$ and define u_{\min} similarly.

Claim 1 *The value of the solution to $\text{MinReg}(K)$ is $V_K = \max\{u_{\max} - u_y, u_z - u_{\min}\}$.*

The following Lemma will be useful in proving Claim 1.

Lemma 1 $V_K < \min\{u_{K-1} - u_{\min}, u_{\max} - u_1\}$.

Proof. I prove that $V_K < u_{K-1} - u_{\min}$. Proving that $V_K < u_{\max} - u_1$ is analogous. Proving that $V_K < u_{K-1} - u_{\min}$ requires proving that (1) $u_z - u_{\min} < u_{K-1} - u_{\min}$ and that (2) $u_{\max} - u_y < u_{K-1} - u_{\min}$. Because $u_z < u_{K-1}$, part (1) follows. To prove part (2), note that $u_{\max} - u_y > u_y - u_{\min}$ because $u_y > t_0$. Thus, if $u_{K-1} > t_0$, then because $u_y > u_{K-1}$ we obtain that $u_{\max} - u_y < u_{\max} - u_{K-1} < u_{K-1} - u_{\min}$ as required. If $u_{K-1} < t_0$, then assume to the contrary that $u_{\max} - u_y > u_{K-1} - u_{\min}$. The last inequality implies that y is u -“closer” than u_{K-1} to the threshold t_0 contradicting the definition of a_{K-1} . Hence, $u_{\max} - u_y \leq u_{K-1} - u_{\min}$ and by assumption (2) the inequality is strict as required. ■

Proof of Claim 1. The history-dependent satisficing procedure of Theorem 3 obtains V_K . I now show that the regret associated with any choice function C that solves $\text{MinReg}(K)$ is at least V_K . Consider the $K + 1$ one-element lists $(a_1), \dots, (a_{K-1}), (y), (z)$. There are several cases to consider:

Case 1. None of these lists is decided. Then, because the informational index is K , two of these lists, (x) and (w) are not separable. Then $C(x, x_{\min}) = C(w, x_{\min}) = x_{\min}$, implying that the regret of C is at least $u_{K-1} - u_{\min} > V_K$, where the inequality follows from Lemma 1.

Case 2. Two or more of these lists are decided. Then the regret of C is at least $u_{\max} - u_1 > V_K$ where the inequality follows from Lemma 1.

Case 3. One of these listed is decided. Then, to obtain V_K , it must be the one-element list (y) . Thus, the regret of C is at least $u_{\max} - u_y$. The remaining K lists are undecided. Because the informational index is K , either two of them are not separable or one of them is not separable from the empty list. In any case, $C(x, x_{\min}) = x_{\min}$ for at least one of these lists, implying that regret is at least $u_z - u_{\min}$. Thus, regret is at least V_K as required. ■

Given an automaton M , denote by $M(L)$ the element M outputs from the list L .

Claim 2 *Let M be an automaton solving $\text{MinReg}(K)$, and let $q_i = g(q_0, a_i)$ for $i \geq 1$. Then,*

(1) *For $i \geq 1$, M remembers a_i : $g(q, a_i) \neq \{q_0, \text{Stop}\}$, if $g(q, x) = q_i$ then either $q = q_i$ or $x = a_i$, $g(q_i, x) \neq q_0$ for $i \geq 1$, and $f(q_i, x) \in \{a_i, x\}$.*

(2) M satisfies in the initial state: for any element $x \neq a_i$, $g(q_0, x) = \text{Stop}$ if $u_x \geq u_y$, or $g(q_0, x) = q_0$ otherwise.

Proof. Let L_0 be the empty list and $L_i = a_i$ for $i \geq 1$. Consider the transition $g(q_k, a_i)$ in M for $0 \leq k \leq K - 1$:

(i) If $g(q_k, a_i) = q_0$ then M reaches q_0 after reading the list (L_k, a_i) . Because $f(q_0, x) = x$, M then chooses x_{\min} from the list (L_k, a_i, x_{\min}) . Thus the regret of M is at least $u_i - x_{\min} > V_K$ in contradiction to M being a solution.

(ii) If $g(q_k, a_i) = \text{Stop}$ then M stops after seeing (L_k, a_i) and thus it chooses either a_i or a_k if $k \neq 0$. But then $\text{regret}_u(M) \geq u_{\max} - u_1 > V_K$ because (x_{\max}) is a possible continuation.

Thus, $g(q_k, a_i) \notin \{\text{Stop}, q_0\}$. Similarly to (i), if $g(q_i, x) = q_0$ and $i \geq 1$ then $M(a_i, x, x_{\min}) = x_{\min}$ and M cannot be a solution. Thus, $g(q_i, x) \neq q_0$ for $i \geq 1$. Clearly, $f(q_i, x) \in \{x, a_i\}$ or else M fails to choose from the list $L = (a_i, x)$ an element that appears in L .

Assume $g(q_k, x) = q_i$ but $q_k \neq q_i$ and $x \neq a_i$. Then M reaches state q_i after reading either (L_k, x) or (a_i) . But then, since $a_i \neq x$ and $a_i \neq a_k$ for $k \geq 1$, we must have $M(a_i, x_{\min}) = M(L_k, x, x_{\min}) = x_{\min}$ and M cannot be a solution.

In particular, $g(q_0, x) \in \{\text{Stop}, q_0\}$ for any $x \neq a_i$. To conclude the proof, note that if $u_x \geq u_y$ but $g(q_0, x) = q_0$ then the regret associated with M is at least $u_x - u_{\min} > u_{k-1} - u_{\min} > V_K$. Similarly, $g(q_0, x) = q_0$ when $u_x \leq u_z$. ■

I now identify an automaton that solves $\text{MinReg}(K)$ subject to u - and *regret*-Efficiency.

Claim 3 *The history-dependent satisficing procedure of Theorem 3 solves $\text{MinReg}(K)$ subject to u - and *regret*-Efficiency.*

Proof. Let M be an automaton implementing the history-dependent satisficing procedure of Theorem 3. Clearly, it solves $\text{MinReg}(K)$. Let q_0 be the initial state of M and denote $q_j = g_M(q_0, a_j)$, where g_A is the transition function of automaton A .

To establish u -Efficiency, assume there exists an automaton M' with K states that is u -superior to M . Then M' solves $\text{MinReg}(K)$ and thus satisfies the conditions of Claim 2. To be u -superior to M , the automaton M' must have at least one transition (not from the initial state) that is different than M . There are three possible cases:

(1) $g_M(q_k, x) \neq \text{Stop}$ but $g_{M'}(q_k, x) = \text{Stop}$. Then $x \neq x_{\max}$ and M outputs a u -larger element from the list (a_k, x, x_{\max}) .

(2) $g_M(q_k, x) = \text{Stop}$ but $g_{M'}(q_k, x) \neq \text{Stop}$. Then, $x \neq a_i$ and $u(x) > (u_k + u_{\max})/2$. Thus M outputs a u -larger element from (a_k, x, x_{\min}) .

(3) $g_M(q_k, x) = q_i$. Then, by (1) $g_{M'}(q_k, x) \neq \text{Stop}$. Thus, $g_{M'}(q_k, x) = q_j$ (where I abuse notation and denote $q_j = g_{M'}(q_0, a_j)$). Because M remembers the u -best element a_t it sees, it must be that $u(a_i) > u(a_j)$. But then $M(a_k, x, x_{\min}) = a_i$ while $M(a_k, x, x_{\min})$ is either a_j or x_{\min} .

To establish *regret*-Efficiency, let the regret of state q_j in M be defined by:

$$p_j = \max\{u_{\max} - u_{y_j}, u_{z_j} - u_j\}$$

where y_j is the u -minimal element for which $g(q_j, y_j) = \text{Stop}$ and z_j the u -maximal element for which $g(q_j, z_j) = q_j$. Then,

Lemma 2 $p_1 \leq p_2 \leq p_{k-1} \leq p_0$.

Proof. I show that $p_{i-1} \leq p_i$ (a similar proof holds for p_{k-1} and p_0). By the definition of M , $u_{y_i} \leq u_{y_{i-1}}$ and thus $u_{\max} - u_{y_i} \geq u_{\max} - u_{y_{i-1}}$. In addition $u_{z_i} \leq u_{z_{i-1}}$. If $u_{z_i} = u_{z_{i-1}}$, then $u_{z_{i-1}} - u_{i-1} < u_{z_i} - u_i$. If $u_{z_{i-1}} > u_{z_i}$ then $u_{z_{i-1}} \geq u_{y_i}$ (because y_i is just above z_i in terms of utility), and thus $u_{\max} - u_{y_i} \geq u_{\max} - u_{z_{i-1}} > u_{z_{i-1}} - u_{i-1}$, where the right inequality follows from $u(z_{i-1}) < u_{i-1}$. Thus $p_{i-1} \leq p_i$. ■

Assume *regret*-Efficiency is violated and let M' be *regret*-superior to M . Then M' solves $\text{MinReg}(K)$ (take L to be the empty list in the definition of *regret*-Efficiency), and thus satisfies the conditions of Claim 2. Let L be a M -undecided list such that $\text{regret}_u(M' | L) < \text{regret}_u(M | L)$.

Assume M reaches state q_i after processing L . Then $\text{regret}_u(M | L) \leq p_i$ because M never moves to state q_j , $j > i$, and because $p_i \geq p_m$ for $m < i$. Since M' solves $\text{MinReg}(K)$, it reaches a state q' , which “remembers” some element a_m . Note that $u(a_m) \leq u_i$ because M remembers the u -highest element from L . Because M' must either ignore or output y_m and z_m , $\text{regret}_u(M' | L) \geq p_m$. Since $p_m \geq p_i$, $\text{regret}_u(M' | L) \geq \text{regret}_u(M | L)$, which is a contradiction. ■

I now show uniqueness, and thus conclude the proof of Theorem 3.

Claim 4 *The unique solution to $\text{MinReg}(K)$ subject to u - and *regret*-Efficiency is the history-dependent satisfying procedure of Theorem 3.*

Proof. Let M' be an automaton that solves $\text{MinReg}(K)$ subject to u - and *regret*-Efficiency. Let M implement the history-dependent satisficing procedure of Theorem 3. Denote $q_i = g_{M'}(q_0, a_i) = g_M(q_0, a_i)$. By u -Efficiency, $f_{M'}(q_i, x)$ must be the u -maximal element among a_i and x , and thus $f_{M'}$ coincides with f_M .

Thus, M' differs from M in some transition $g(q_i, x)$ where $q_i \neq q_0$. To see that this is impossible, I will show that a difference in transitions implies that M is *regret*-superior to M' . Note that by Claim 3, $\text{regret}_u(M \mid L) \leq \text{regret}_u(M' \mid L)$ for every list which is M - and M' -undecided, and thus it is enough to show a strict improvement for just one list. Define y_i, z_i and p_i as in Claim 3, and consider the following cases.

Case 1. $x \notin \{a_1, \dots, a_{k-1}\}$, $u_x > \frac{u_{\max} + u_i}{2}$ but $g_{M'}(q_i, x) = q_i$.

Then, $\text{regret}_u(M' \mid (a_i)) \geq u_x - u_i$ (e.g. consider the list (a_i, x, x_{\min})). But $u_x - u_i > u_{\max} - u_x \geq u_{\max} - u_{y_i}$ by the definition of x and y_i , and $u_x - u_i > u_{z_i} - u_i$ by the definition of z_i . Thus, $u_x - u_i > \max\{u_{\max} - u_{y_i}, u_{z_i} - u_i\} = p_i = \text{regret}_u(M \mid (a_i))$. Thus, $\text{regret}_u(M' \mid (a_i)) > \text{regret}_u(M \mid (a_i))$, contradicting *regret*-Efficiency.

Case 2. $x \notin \{a_1, \dots, a_{k-1}\}$, $u_x < \frac{u_{\max} + u_i}{2}$ but $g_{M'}(q_i, x) = \text{Stop}$.

Then, similarly to case 1,

$$\text{regret}_u(M' \mid (a_i)) \geq u_{\max} - u_x > \max\{u_{\max} - u_{y_i}, u_{z_i} - u_i\} = p_i = \text{regret}_u(M \mid (a_i)).$$

By cases 1 and 2, M and M' output and ignore the same elements in state q_i . For the remaining cases assume $x = a_j$ for $j \geq 1$.

Case 3. $g_{M'}(q_i, a_j) \neq g_M(q_i, a_j)$ and $p_i \neq p_j$.

Again, this implies *regret*-Efficiency is violated. For example, if $g_M(q_i, a_j) = q_i$ and $g_{M'}(q_i, a_j) = q_j$ then $u_j < u_i$, and thus $\text{regret}_u(M' \mid (a_i, a_j)) \geq p_j > p_i = \text{regret}_u(M \mid (a_i, a_j))$.

By case 3, M' differs from M only in transitions of the form $g_{M'}(q_i, a_j)$ for which $p_i = p_j$, and at least one such transition exists. This implies that M' violates u -Efficiency.

Indeed, if $g_{M'}(q_i, a_j) \neq g_M(q_i, a_j)$ then M outputs a u -larger element from the list (a_i, a_j, x_{\min}) . In addition, consider any other list L . When reading L , the transitions of M' and M are identical until the first time in which an element a_t appears such that $q_l = g_{M'}(q_r, a_t) \neq g_M(q_r, a_t) = q_h$ where $l, h \in \{r, t\}$ and $u_l < u_h$. The two automata then move to different states in which they make the same decisions for elements $x \in X \setminus \{a_1, \dots, a_{k-1}\}$. For an element a_m , there are several possibilities:

(1) $g_{M'}(q_l, a_m) \neq g_M(q_l, a_m)$ where q_l is the current state of M' . Then $p_m = p_l = p_h$ and the two automata continue to make the same decisions regarding elements in $x \in X \setminus \{a_1, \dots, a_{K-1}\}$.

(2) $g_{M'}(q_l, a_m) = g_M(q_l, a_m)$. Then there are two possibilities:

(2.1) $u_m \geq u_h$. Then M and M' move to the same next state q_m .

(2.2) $u_m < u_h$. Then M' moves to state q_s ($s \in \{l, m\}$) while M stays in state q_h . Since this transition in M' is identical to the corresponding transition in M it must be that $p_s \leq p_l = p_h$. But since $u_s < u_h$ (or else M would move to state q_s as well) it must also be that $p_s \geq p_h$. Thus $p_s = p_h$, which means M and M' continue to make the same decisions (except maybe when the list ends in which case M outputs a weakly u -higher element).

Thus, M is u -superior to M' as required. ■

References

- [1] Dilip Abreu and Ariel Rubinstein, *The structure of nash equilibrium in repeated games with finite automata*, *Econometrica* **56** (1988), no. 6, 1259–1281.
- [2] Gordon W. Allport, *The nature of prejudice*, Perseus Books Publishing, 1954.
- [3] Taradas Bandyopadhyay, *Revealed preference theory, ordering and the axiom of sequential path independence*, *Review of Economic Studies* **55** (1988), no. 2, 343–351.
- [4] Colin Camerer, *Behavioral economics*, *Advances in Economics and Econometrics: Theory and Applications* (Richard Blundell, Whitney Newey, and Torsten Persson, eds.), vol. 2, *Econometric Society Monographs*, 2006, pp. 181–214.
- [5] Donald E. Campbell, *Realization of choice functions*, *Econometrica* **46** (1972), no. 1, 171–180.
- [6] John F. Dovidio, Peter S. Glick, and Laurie A. Rudman, *On the nature of prejudice: Fifty years after allport*, Blackwell Publishing, New York, 2005.
- [7] James Dow, *Search decisions with limited memory*, *The Review of Economic Studies* **58** (1991), no. 1, 1–14.
- [8] Kfir Eliaz, *Nash equilibrium when players account for the complexity of their forecasts*, *Games and Economic Behavior* **44** (2003), no. 2, 286–310.

- [9] Ronald Fryer and Matthew O. Jackson, *A categorical model of cognition and biased decision making*, The B.E. Journal of Theoretical Economics: Contributions **8** (2008), no. 1, Article 6.
- [10] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to automata theory: Languages and computation*, Addison Wesley, Cambridge, Massachusetts, 1979.
- [11] Sheena S. Iyengar and Mark R. Lepper, *When choice is demotivating: Can one desire too much of a good thing*, Journal of Personality and Social Psychology **79** (2000), no. 6, 995–1006.
- [12] Daniel Kahneman, Jack L. Knetsch, and Richard H. Thaler, *Anomalies: The endowment effect, loss aversion, and status quo bias*, Journal of Economic Perspectives **5** (1991), no. 1, 193–206.
- [13] Daniel Kahneman and Amos Tversky, *Prospect theory: An analysis of decision under risk*, Econometrica **47** (1979), no. 2, 263–291.
- [14] Ehud Kalai and William Stanford, *Finite rationality and interpersonal complexity in repeated games*, Econometrica **56** (1988), no. 2, 397–410.
- [15] Gil Kalai, *Learnability and rationality of choice*, Journal of Economic Theory **113** (2003), no. 1, 104–117.
- [16] Barton L. Lipman and Sanjay Srivastava, *Informational requirements and strategic complexity in repeated games*, Games and Economic Behavior **2** (1990), no. 3, 273–290.
- [17] Carolyn B. Mervis and Eleanor Rosch, *Categorization of natural objects*, Annual Review of Psychology **32** (1981), 89–115.
- [18] Abraham Neyman, *Bounded complexity justifies cooperation in the finitely repeated prisoner’s dilemma*, Economic Letters **19** (1985), 227–229.
- [19] Eleanor Rosch, *Principles of categorization*, Cognition and categorization (Eleanor Rosch and B. B. Lloyd, eds.), Erlbaum, Hillsdale, NJ, 1978.
- [20] Ariel Rubinstein, *Finite automata play the repeated prisoners’ dilemma*, Journal of Economic Theory **39** (1986), 83–96.
- [21] ———, *Why are certain properties of binary relations relatively more common in natural language*, Econometrica **64** (1996), 343–356.

- [22] ———, *Modeling bounded rationality*, Zeuthen Lecture Book Series, The MIT Press, Cambridge, Massachusetts, 1998.
- [23] Ariel Rubinstein and Yuval Salant, *A model of choice from lists*, *Theoretical Economics* **1** (2006), no. 1, 3–17.
- [24] William Samuelson and Richard Zeckhauser, *Status quo bias in decision making*, *Journal of Risk and Uncertainty* **1** (1988), 7–59.
- [25] Herbert A. Simon, *A behavioral model of rational choice*, *Quarterly Journal of Economics* **69** (1955), 99–118.
- [26] Amos Tversky and Daniel Kahneman, *Loss aversion in riskless choice: a reference-dependent model*, *Quarterly Journal of Economics* **106** (1991), no. 4, 1039–1061.
- [27] Andrea Wilson, *Bounded memory and biases in information processing*, *NAJ Economics* **5:3** (2002).